Better path-finding algorithms in LPS Ramanujan graphs

Carvalho Pinto, Eduardo; Petit, Christophe

DOI:
10.1515/jmc-2017-0051

Document Version
Peer reviewed version

Citation for published version (Harvard):
Carvalho Pinto, E & Petit, C 2018, 'Better path-finding algorithms in LPS Ramanujan graphs', *Journal of Mathematical Cryptology*, vol. 12, no. 4, pp. 191-202. https://doi.org/10.1515/jmc-2017-0051

Noname manuscript No.
(will be inserted by the editor)

Better path-finding algorithms in LPS Ramanujan graphs

Eduardo Carvalho Pinto · Christophe Petit

Abstract We provide a new heuristic polynomial time algorithm that computes short paths between arbitrary pairs of vertices in Lubotzky-Philipps-Sarnak's Ramanujan graphs. The paths returned by our algorithm are shorter by a factor approximately 16/7 compared to previous work, and they are close to optimal for vertices corresponding to diagonal matrices. Our results also lead to an improved cryptanalysis of Charles-Goren-Lauter hash function.

Mathematics classification and key words: 94A60; 11Y50; 05C12; 05C25; 08A50; cryptography; Cayley graph; Tillich-Zémor hash function; path finding algorithm.

1 Introduction

LPS graphs were introduced by Lubotzky, Philipps and Sarnak [8]. LPS graphs are Ramanujan graphs, namely they are graphs with optimal expansion properties, and they have many applications in mathematics and computer science [7]. In [2], Charles, Goren and Lauter introduced a cryptographic hash function whose security relied on the presumed computational intractability of computing short paths between arbitrary pairs of vertices in the graph. Due to the algebraic nature of LPS graphs this problem can also be stated in group-theoretic terms: given the group $G = PSL(2, \mathbb{F}_p)$ and a particular set of generators, and given an element of G, find a short factorization of this element as a product of the generators. This problem was later solved in [15, 10], leading to a full cryptanalysis of the cryptographic construction.

The quality of a path-finding algorithm is measured by its running time, but also by the lengths of the paths returned. In this paper we are interested in minimizing this length while keeping a polynomial run time complexity. While the algorithm in [10] returns paths that are a priori optimal up to a constant factor 16/3, we improve this algorithm and decrease this factor down to 7/3 under several plausible heuristic conditions. Similarly to [10] we first solve the case of diagonal matrices, then we reduce the general case to this case. Our algorithm improves both the diagonal case and the reduction, and for the diagonal case it returns factorizations with lengths very close to the expected minimal.

LPS graphs have recently been suggested for quantum circuit design. In this setting LPS generators correspond to elementary quantum gates, and they are combined to approximate arbitrary quantum gates on a single qbit. Interestingly, an algorithm similar to our diagonal algorithm has been independently developed in that context [1, 11]. It was later transposed to our setting [12], resulting in an algorithm very similar to our algorithm in the diagonal case. It seems very likely that our other improvement (reducing from the general case to the diagonal case) can also be adapted to the problem of quantum circuit design; this will be the purpose of a future paper.

1.1 Outline

The paper is organized as follows. Section 2 describes LPS graphs and previous path-finding algorithms for these graphs; Section 3 gives an optimal algorithm when the optimal solution is short; Section 4 describes the diagonal case; Section 5 gives our new reduction and describe the full algorithm; Section 6 gives experimental support to our heuristic analysis and Section 7 concludes the paper.

Eduardo Carvalho, Ecole Polytechnique · Christophe Petit, University of Birmingham, E-mail: christophe.f.petit@gmail.com, ORCID 0000-0003-3482-6743

2 Path-finding algorithms in LPS Ramanujan graphs

In this section we recall the definition of LPS graphs and we briefly describe previous path-finding algorithms in these graphs. In our exposition we mostly follow [15,10], and we refer to those papers for details and proofs.

2.1 LPS Ramanujan graphs

In this paper p and ℓ will be two distinct primes congruent to 1 modulo 4 such that ℓ is a quadratic residue modulo p. We think of p and ℓ as being respectively "large" and "small" primes, and in fact in our complexity estimates we will assume $\ell = O(1)$. We denote the finite field with p elements by $\mathbb{F}_p$. With an abuse of notation we will often identify elements of $\mathbb{F}_p$ and integers in $\{0, 1, \ldots, p-1\}$. For any ring R, we write $GL_2(R)$ and $PSL_2(R)$ respectively for the general linear group and the projective special linear group of rank 2 over R. We define $G := PSL_2(\mathbb{F}_p)$. We write I for the identity matrix in any of those matrix groups.

We denote by $B = \mathbb{Q}[i,j]$ the quaternion algebra over the rationals generated by two elements i and j such that $i^2 = j^2 = -1$ and $k := ij = -ji$. For any $q = a + bi + cj + dk \in B$, the conjugate of q is $\overline{q} := a - bi - cj - dk$ and the norm of q is $n(q) = q\overline{q} = a^2 + b^2 + c^2 + d^2$. Note that the norm is multiplicative: for any $q_1, q_2 \in B$, we have $n(q_1 q_2) = n(q_1)n(q_2)$.

With some abuse of notation we also use the symbol i for the imaginary unit, and we denote the Gaussian integers by $\mathbb{Z}[i]$. The map $\sigma : GL_2(\mathbb{Z}[i]) \to B$ defined by

$$m = \begin{pmatrix} a+bi & c+di \\ -c+di & a-bi \end{pmatrix} \;\to\; \sigma(m) = a + bi + cj + dk$$

is an isomorphism of algebras. Note that we have $\det m = n(\sigma(m))$, and that both σ and its inverse are efficiently computable. For any matrix $m = \left(\begin{smallmatrix} a+bi & c+di \\ -c+di & a-bi \end{smallmatrix}\right) \in GL_2(\mathbb{Z}[i])$ we define its conjugate matrix by $\overline{m} := \left(\begin{smallmatrix} a-bi & -c-di \\ c-di & a+bi \end{smallmatrix}\right)$. Clearly, we have $\overline{\overline{m}} = m$ and $\sigma(\overline{m}) = \overline{\sigma(m)}$.

Following [15] for any integer $e \geq 1$ we define E_e as the set of 4-tuples $(a,b,c,d) \in \mathbb{Z}^4$ such that

$$\begin{cases} a^2 + b^2 + c^2 + d^2 = \ell^e, \\ a > 0,\ a = 1 \bmod 2, \\ b = c = d = 0 \bmod 2. \end{cases}$$

Up to multiplication by a unit there are $\ell + 1$ elements of norm ℓ in B; these correspond to elements of E_1. We also denote by Σ the set of matrices corresponding to E_1, and we define

$$\Omega := \left\{ \begin{pmatrix} a+bi & c+di \\ -c+di & a-bi \end{pmatrix} | (a,b,c,d) \in E_e \text{ for some integer } e > 0 \right\}.$$

Note that Σ is symmetric in the sense that for any $s \in \Sigma$, there exists $s' = \overline{s} \in \Sigma$ with $ss' = \ell I$. An important observation for the path-finding algorithms below is that matrices in Ω admit essentially unique factorizations in the elements of Σ.

Lemma 1 ([15], citing [8,5,14]) *Any matrix in Ω can be expressed in a unique way as a product*

$$M = \pm \ell^r M_1 M_2 \ldots M_e$$

where $\log_\ell(\det(M)) = e + 2r$ and $M_i \in \Sigma$ and $M_i M_{i+1} \neq \ell I$ for $i = 1, \ldots, e-1$.

LPS graphs were introduced by Lubotsky, Philips and Sarnak in [8]. Let $\iota \in \mathbb{F}_p$ such that $\iota^2 = -1$. Reduction modulo p extends into a group homomorphism $\phi : GL_2(\mathbb{Z}[i]) \to G$ defined by

$$\phi \begin{pmatrix} a+bi & c+di \\ -c+di & a-bi \end{pmatrix} = \begin{pmatrix} a+b\iota & c+d\iota \\ -c+d\iota & a-b\iota \end{pmatrix}.$$

Let $S = \{\phi(s) | s \in \Sigma\}$ be the set of images of elements in Σ through the homomorphism ϕ. The LPS graph for parameters p and ℓ is the Cayley graph constructed from the group G and the generator set S; in other words it is a graph whose vertices correspond to the elements of G, and such that there is an edge between two vertices corresponding to g_1 and g_2 if and only if there is an element $s \in S$ such that $g_2 = g_1 s$. This graph is an undirected $(\ell+1)$-regular graph. For any fixed ℓ and increasing p, LPS graphs form a family of Ramanujan graphs, in other words they are optimal expander graphs [8,7].

LPS graphs were used by Charles, Goren and Lauter in [3] to construct families of cryptographic hash functions. The security arguments for the construction relied on the hope that there was no efficient algorithm to solve the following computational problem.

Problem 1 *Let p, ℓ, G, S as above, and let $m \in G$. Write m as a "short" product of elements in S.*

Recommended parameters in [3] are $\ell = 5$ and p a 1024-bit prime. For practical purposes "short" could be given a concrete value such as 2^{40}; in this paper we consider a product as short when it is made of at most L products, where L is a polynomial function of $\log p$. This is consistent with the definition of Babai's conjecture on the diameter of simple groups, as formulated by Helfgott in [6].

2.2 Finding paths in LPS graphs

In this paper, we are interested in algorithms to solve Problem 1, or equivalently to compute short paths between any two vertices in LPS graphs.

Let all notations be as above, and let m be a matrix in G that we want to write as a short product of elements from the set S.

The algorithm of Petit et al. [10] first decomposes the matrix m as a product

$$m = \lambda \cdot D_1 \cdot s \cdot D_2 \cdot s \cdot D_3 \cdot s \cdot D_4 \tag{1}$$

where $\lambda \in \mathbb{F}_p^*$, the factors D_i are diagonal matrices with a non zero square determinant, and s is a particular (arbitrary) generator in the set S. As the equation is over the projective special linear group, each diagonal matrix can be normalized as $D_i = \left(\begin{smallmatrix} 1 & 0 \\ 0 & \alpha_i \end{smallmatrix}\right)$. Equation 1 then amounts to a small polynomial system with four equations and five variables. The algorithm given in [10] to solve this system picks random solutions until all α_i are square, and it is heuristically expected to need 16 trials on average.

Petit et al. [10] also provide an algorithm to factor any diagonal matrix with square determinant in a short product of the generators S. This algorithm extends a previous algorithm from Tillich and Zémor [15] for the identity matrix. First the diagonal matrix is lifted into an element of Ω, then a factorization of this element as a product of the elements of Σ is computed. The factorization of the input diagonal matrix in the elements of $S = \phi(\Sigma)$ follows by the group homomorphism ϕ. We now give some details on the first and second step.

Let $m = \left(\begin{smallmatrix} A+B\iota & 0 \\ 0 & A-B\iota \end{smallmatrix}\right) \in G$ be a diagonal matrix with $\det m = A^2 + B^2$ a square. The lifting step consists in finding $e \in \mathbb{N}$ and $\lambda, w, x, y, z \in \mathbb{Z}$ with

$$\begin{cases} (A\lambda + wp)^2 + (B\lambda + xp)^2 + 4p^2(y^2 + z^2) = \ell^e \\ A\lambda + wp = 1 \bmod 2, \\ B\lambda + xp = 0 \bmod 2. \end{cases} \tag{2}$$

After fixing e large enough, Petit et al. solve the norm equation modulo p; as $A^2 + B^2$ is a square this gives two possible values for λ and one is picked randomly. Next, the norm equation is considered modulo p^2: this gives a bilinear equation in w and x, and a random solution is selected. At this point, the norm equation is considered over the integers, and it gives

$$4(y^2 + z^2) = n$$

where

$$n := \left(\ell^e - (A\lambda + wp)^2 - (B\lambda + xp)^2\right)/p^2 \in \mathbb{Z} \tag{3}$$

for λ, x, z chosen as before. This equation has a solution whenever $n/4$ is an integer, and all prime factors of n congruent to 3 modulo 4 appear an even number of times in the factorization of n. To avoid a costly factorization step, Petit et al. suggest to pick random solutions for (w, x) until n is 4 times a prime congruent to 1 modulo 4. Suitable values for (y, z) are then computed with Cornacchia's algorithm [4]. Taking e larger than $\log 8p^4 \approx 4 \log p$ ensures that n is positive in this algorithm.

Once the integers e, λ, w, x, y, z are found they give a matrix

$$\tilde{m} = \begin{pmatrix} (A+wp)+(B+xp)i & 2yp+2zpi \\ -2yp+2zpi & (A+wp)-(B+xp)i \end{pmatrix} \in \Omega$$

reducing to m modulo p. Remember that by Lemma 1 factorization is essentially unique in Ω. Tillich and Zémor showed in [15] how to recover all the factors successively: for any $s \in \Sigma$ we have $\tilde{m} = \tilde{m}'s$ with $\tilde{m}' \in \Omega$ if and only if $\tilde{m}s^{-1} \in \Omega$.

The correctness and running time of this algorithm are analyzed in the following lemma:

Lemma 2 *For any $\tilde{m} \in \Omega$, Algorithm 1 successively recovers the factorization given by Lemma 1, in a time $O(\log_\ell^2 \det \tilde{m})$.*

Algorithm 1 Factorization in Ω

Require: $\tilde{m} \in \Omega$
Ensure: list $L = (s_1, \ldots, s_N)$ such that $s_i \in \Sigma$ and $\tilde{m} = \prod_i s_i$
1: Initialize L to an empty list
2: Find largest r such that ℓ^r divides all coefficients of $\tilde{m}$
3: $\tilde{m} \leftarrow \tilde{m}/\ell^r$
4: **while** $\det(\tilde{m}) \neq 1$ **do**
5: Find $s \in \Sigma$ such that ℓ divides all coefficients of $\tilde{m}\overline{s}$
6: $\tilde{m} \leftarrow \tilde{m}\overline{s}/\ell$
7: Append s at the beginning of list L
8: **end while**
9: **return** L

Proof Let $\tilde{m} \in \Omega$ and $\det(\tilde{m}) \neq 1$. Then, there is a unique $s \in \Sigma$ such that $\tilde{m} = \tilde{m}'s$ with $\tilde{m}' \in \Sigma$, by Lemma 1. Therefore, there is an equally unique $s' = \overline{s} \in \Sigma$ such that $ss' = \ell I$ and $\tilde{m}'ss'$ is divisible by ℓ. . This way, by iterating over Σ, the algorithm always finds the matrix s if $\det(\tilde{m}) \neq 1$, adding it to the beginning of the list $\mathcal{L}$, and finishes when $\det(\tilde{m}) = 1$. In the end, $\mathcal{L}$ holds the complete factorization in the right order for $\tilde{m}$ on elements of Σ.

As for the running time, one can write all coefficients of $\tilde{m}$ in base ℓ in $O(\log_\ell \det \tilde{m})$ bit operations, after which Step 3 becomes trivial. The while loop is executed $\log_\ell \det \tilde{m}$ times and each iteration requires $O(\ell^2) = O(1)$ multiplications of a $O(\log_\ell \det \tilde{m})$ length element by a $O(\ell) = O(1)$ length element, and each of them requires $O(\log_\ell \det \tilde{m})$ bit operations.

The algorithm of this paper will incorporate two crucial improvements over Petit et al.'s algorithm to reduce factorization lengths from $16 \log_\ell p$ to $(7 + o(1)) \log_\ell p$. In Section 4 we will reduce the factorization lengths for diagonal matrices from $4 \log_\ell p$ to $(3 + o(1)) \log_\ell p$. In Section 5, we show how to factor arbitrary matrices with two diagonal matrices plus a third matrix with factorization length $\log_\ell p$.

We will provide an alternative method to reduce the general factorization problem to the factorization of only two diagonal matrices instead of four.

3 Algorithm for short paths

Our first algorithm computes the shortest path between any two matrices when the length of this shortest path is short enough. This is useful in a cryptanalysis of Charles-Goren-Lauter hash function assuming short messages are hashed.

The algorithm relies on the following simple observation: when the products are short, then no reduction modulo p occurs and the lifting step required in previous algorithms [10,15] becomes trivial.

Lemma 3 *Let m be a product of at most $\lfloor 2 \log_l(p/2) \rfloor$ matrices in Σ. Then there exist integers $A, B, C, D \in \left[-\lceil \frac{p}{2} \rceil, \lfloor \frac{p}{2} \rfloor\right]$ such that $m = \begin{pmatrix} A+Bi & C+Di \\ -C+Di & A-Bi \end{pmatrix}$.*

Proof Let $m = m_1 \ldots m_k = \begin{pmatrix} A+Bi & C+Di \\ -C+Di & A-Bi \end{pmatrix}$ be a product of k matrices in Σ. Then, we know that

$$\det m = A^2 + B^2 + C^2 + D^2 = \ell^k.$$

If we choose k to be at most $2 \log_l(p/2)$ then

$$A^2 + B^2 + C^2 + D^2 = \ell^k \leq \ell^{2 \log_l(p/2)} = \frac{p^2}{4}.$$

This implies that $A, B, C, D \in \left[-\lceil \frac{p}{2} \rceil, \lfloor \frac{p}{2} \rfloor\right]$.

More generally, let $m_1, m_2 \in G$ such that there exists a product m of at most $\lfloor 2 \log_l(p/2) \rfloor$ matrices in $\phi(\Sigma)$ such that $m_2 = m_1 m$. For every possible length e up to $\lfloor 2 \log_l(p/2) \rfloor$, Algorithm 2 normalizes the matrix to ensure its determinant is ℓ^e, then it writes the result in the form $\begin{pmatrix} a+b\iota & c+d\iota \\ -c+d\iota & a-b\iota \end{pmatrix}$ and it checks whether a, b, c, d are small enough to be lifted directly to Ω. When the correct factorization length is identified the algorithm then proceeds as in Tillich-Zémor algorithm to compute all the factors.

We remark that there is at most one value of $e \leq \lfloor \log_l(p) \rfloor$ so that the algorithm succeeds: indeed the girth of LPS graphs is larger than $2 \log_\ell p$ when ℓ is a quadratic residue modulo p [8]. Lemma 4 below gives the complexity of this algorithm.

Algorithm 2 Short path algorithm

Require: $m_1, m_2 \in G$ such that $m_1 = m_2 \prod_{i=1}^{N} s_i$ with $N \leq \lfloor 2\log_l(p/2) \rfloor$
Ensure: list of factors $L = (s_1, \ldots, s_N)$
1: $m \leftarrow m_2^{-1} m_1$
2: Normalize the representation of m so that it has determinant 1
3: **for** $e \in \{1, \ldots, \lfloor 2\log_l(p/2) \rfloor\}$ **do**
4: Write $m\sqrt{\ell^e}$ in the form $\left(\begin{smallmatrix} a+b\iota & c+d\iota \\ -c+d\iota & a-b\iota \end{smallmatrix}\right)$ with $a, b, c, d \in (-\frac{p}{2}, \frac{p}{2})$
5: **if** $a^2 + b^2 + c^2 + d^2 = \ell^e$ (over the integers) **then**
6: **go to** Step 9
7: **end if**
8: **end for**
9: Run Algorithm 1 on $\tilde{m} := \left(\begin{smallmatrix} a+bi & c+di \\ -c+di & a-bi \end{smallmatrix}\right)$ to get a list of factors L
10: Replace every element s in L by its projection $\phi(s)$
11: **return** L

Lemma 4 *Let $m_1, m_2 \in G$ such that $m_2^{-1} m_1$ has a factorization of length $e \leq \lfloor 2\log_\ell(p/2) \rfloor$ in the elements of $\phi(\Sigma)$. Then Algorithm 2 returns the smallest such factorization using $\tilde{O}(\log_\ell^2 p)$ bit operations.*

Proof Step 2 requires to compute a determinant, a square root modulo p and a division modulo p. One can also precompute the square root of ℓ modulo p and the matrix $m\sqrt{\ell}$ between Step 2 and 3, and compute $m\sqrt{\ell^e}$ in Step 4 as $\ell \cdot (m\sqrt{\ell^{e-2}})$. Given the matrix $m\sqrt{\ell^e} = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$, one can complete Step 4 by computing $a = \frac{A+D}{2} \bmod p$, $b = \frac{A-D}{2i} = -i\frac{A-D}{2}$, etc. This requires a few divisions by 2 and a few full multiplications modulo p. As the cost of a square root is $O(\log p)$ multiplications, and as multiplying and dividing by small integers requires a few additions, the cost of the whole algorithm up to Step 8 is bounded by $O(\log p)$ multiplications, which is $\tilde{O}(\log^2 p)$ bit operations. By Lemma 2 the last step is faster than this, so the whole algorithm costs $\tilde{O}(\log^2 p)$ bit operations.

4 Improved algorithm for diagonal matrices

We now turn to diagonal matrices, and improve the algorithm of Petit et al. [10] to return factorizations of length $(3 + o(1)) \log p$ instead of $4 \log p$. A similar algorithm was independently suggested in [12].

Let $m = \left(\begin{smallmatrix} A+B\iota & 0 \\ 0 & A-B\iota \end{smallmatrix}\right) \in G$ be a diagonal matrix with determinant $(A^2 + B^2)$ a quadratic residue modulo p. If $B = 0$ then m is the identity matrix and it admits a trivial factorization. So we now assume $B \neq 0$.

Our algorithm follows the lines of Petit et al.'s algorithm sketched in Section 2.2. Recall that in this algorithm we need $e \approx 4 \log p$ to ensure that the value n defined by Equation 3 is positive. In their algorithm (w, x) is chosen as a random solution to the norm equation modulo p^2 such that n is of a form that makes Cornacchia's algorithm efficient. Our main idea for the algorithm of this section is to choose (w, x) not randomly but

1. to minimize the value $(A\lambda + wp)^2 + (B\lambda + xp)^2$,
2. while keeping $n = \left(\ell^e - (A\lambda + wp)^2 - (B\lambda + xp)^2\right)/p^2$ of a form that makes Cornacchia's algorithm efficient.

We now show how to model the minimization problem as a variant of a closest vector problem in a two dimensional lattice, which can then be solved using classical techniques such as lattice reduction, the Euclidean algorithm, or equivalently continued fractions.

Let e be a positive integer (to be fixed later) and let $\lambda \in \mathbb{Z}$ such that $\lambda^2(A^2 + B^2) = \ell^e \bmod p$. Let $w_0 = 1$ if $A\lambda$ is even and $w_0 = 0$ otherwise. Similarly, let $x_0 = 0$ if $B\lambda$ is even and $x_0 = 1$ otherwise. For any couple (x, y) that is part of a solution to System 2 there corresponds a couple of integers (w', x') such that $w = w_0 + 2w'$ and $x = x_0 + 2x'$. Moreover we have

$$(A\lambda + (w_0 + 2w')p)^2 + (B\lambda + (x_0 + 2x')p)^2 = \ell^e \bmod p^2$$

or equivalently

$$\alpha w' + x' = \beta \bmod p \tag{4}$$

where we defined

$$\alpha := A/B \bmod p, \qquad \beta := \left(\frac{\ell^e - \lambda^2(A^2 + B^2)}{p} - 2\lambda(Aw_0 + Bx_0)\right)/4\lambda B \bmod p.$$

We observe that the solutions to Equation 4 are obtained by translation of a certain lattice.

Lemma 5 *The set of solutions $(w', x') \in \mathbb{Z}^2$ to Equation 4 is the set $\mathcal{L} + (0, \beta) = \{v + (0, \beta) \mid v \in \mathcal{L}\}$, where $\mathcal{L}$ is the two-dimensional lattice generated by the vectors $(1, -\alpha)$ and $(0, p)$.*

We then re-write the norm we want to minimize as the distance between a particular element in the Euclidean plane and a lattice element:

$$(A\lambda + (w_0 + 2w')p)^2 + (B\lambda + (x_0 + 2x')p)^2 = 4p^2 \, ||v - t||^2$$

where $v := (w', x') - (0, \beta)$ is a lattice point and $t := \left(-\frac{A\lambda + w_0 p}{2p}, -\frac{B\lambda + x_0 p}{2p} - \beta\right)$. Minimizing this norm is an instance of the *closest vector problem* for which standard algorithmic solutions exist [9]. We adapt these solutions to incorporate the additional requirement that $n := \ell^e - 4p^2 \, ||v - t||^2$ is of a form that makes Cornacchia's algorithm efficient.

Our strategy is to first compute a minimal basis for the lattice, namely two vectors that generate the lattice and have minimal norms. This can be done using Gauss reduction, or equivalently the Euclidean algorithm. We then write t as a linear combination of the two short vectors, and we round the coefficients to obtain a lattice vector v that is close to t. Finally, we add small lattice vectors to v and compute the corresponding value of n, until it is of a suitable form. Once we find n of a suitable form, we compute the values of w and x and we apply Cornacchia's algorithm to compute y and z. In our algorithm we start with an a priori sufficient value $e = 3 \log_\ell p + \log \log_\ell p$, determined by our complexity analysis below, and then we increase e during the course of the algorithm if no n value of a suitable form can be found. This strategy is described in Algorithm 3.

Algorithm 3 Diagonal matrix algorithm

Require: a diagonal matrix $D \in G$
Ensure: list of factors $\mathcal{L} = (s_1, \ldots, s_N)$ such that $s_i \in S$ and $D = \prod_{i=1}^{N} s_i$

1: Write D as $D = \begin{pmatrix} A+B\iota & 0 \\ 0 & A-B\iota \end{pmatrix}$
2: **if** $B = 0$ **then**
3: **return** a void list $\mathcal{L}$
4: **end if**
5: $e \leftarrow \lceil 3 \log_\ell p + \log \log_\ell p \rceil$
6: $\lambda \leftarrow$ square root of $\ell^e/(A^2 + B^2)$ modulo p
7: $(w_0, x_0) \leftarrow (1 - A\lambda, B\lambda) \bmod 2$
8: $\alpha \leftarrow A/B \bmod p$
9: $\beta \leftarrow \left(\frac{\ell^e - \lambda^2(A^2+B^2)}{p} - 2\lambda(Aw_0 + Bx_0)\right)/4\lambda B \bmod p$
10: $\mathcal{L} \leftarrow$ lattice generated by $(1, \beta)$ and $(0, p)$
11: $\{v_1, v_2\} \leftarrow$ reduced basis for $\mathcal{L}$
12: $t \leftarrow (-A\lambda/2p - w_0/2, -B\lambda/2p - x_0/2 - \beta)$
13: $v \leftarrow$ element in $\mathcal{L}$ closest to t
14: $(w, x) = (w_0, x_0) + 2v + 2(0, \beta)$
15: $n \leftarrow \frac{\ell^e - (A\lambda + wp)^2 - (B\lambda + xp)^2}{4p^2}$
16: **if** $n > 0$ **then**
17: **if** n is "nice for Cornacchia's algorithm"
18: (for example n is a (pseudo)prime and $n = 1 \bmod 4$) **then**
19: $(y, z) \leftarrow$ solution to norm equation $y^2 + z^2 = n$
20: **else**
21: $v = (w', x') \leftarrow$ element in $\mathcal{L}$ closest to t not yet used
22: **go to** Step 14
23: **end if**
24: **else**
25: $e \leftarrow e + 1$
26: **go to** Step 6
27: **end if**
28: $\tilde{m} \leftarrow \begin{pmatrix} (A+wp)+(B+xp)i & 2yp+2zpi \\ -2yp+2zpi & (A+wp)-(B+xp)i \end{pmatrix}$
29: Run Algorithm 1 on $\tilde{m}$ to get a list of factors $\mathcal{L}$
30: Replace every element s in $\mathcal{L}$ by its projection $\phi(s)$
31: **return** $\mathcal{L}$

While the algorithm of Petit et al. returns factorizations of length $4 \log_\ell p$, we expect our algorithm to return factorizations of length $(3 + o(1)) \log_\ell p$ on most inputs. Our analysis in the proof of the following lemma relies on the probability that the number n generated by the algorithm is a prime. We approximate this by the probability that random numbers of the same size are prime. The bound only holds for inputs such that the corresponding lattice satisfies the so-called "Gaussian heuristic", namely when the two vectors in a minimal basis are of norm roughly $\sqrt{\det \mathcal{L}}$. This will be true for most inputs, but for exceptional inputs one of these vectors may be significantly larger than the other one, in which case the best achievable bound

is $(4 + o(1)) \log_\ell p$. We refer to [12] for a more thorough analysis on a very similar algorithm, and to [13] for some progress on the heuristic assumptions involved in this analysis.

Lemma 6 *Let $m \in G$ be a diagonal matrix, not equal to the identity. Then under plausible heuristic assumptions Algorithm 3 requires $\tilde{O}(\log^3 p)$ bit operations and for most inputs it returns factorizations of lengths $e = (3 + o(1)) \log_\ell p$.*

Proof Correctness of the algorithm follows from our description above; we therefore focus on expected length and running time.

The lattice $\mathcal{L}$ has discriminant p, therefore by the Gaussian heuristic we can expect that it contains a basis of vectors of length approximately $p^{1/2}$. As a result, the minimal value of $(A\lambda + wp)^2 - (B\lambda + xp)^2$ satisfying the norm equation modulo p^2 is of size approximately p^3, and we must take e at least $3 \log_\ell p$. This minimal value, however, may not lead to an n of the correct form for Cornacchia's algorithm, so our algorithm tries other short vectors in the lattice. Following estimates given by the prime number theorem we heuristically expect that $O(\log p^3) = O(\log p)$ trials will be necessary and sufficient. As a result we can correct our estimate for $(A\lambda + wp)^2 - (B\lambda + xp)^2$ to approximately $p^3 \log p$, and expect that $e \approx 3 \log_\ell p + \log \log_\ell p$ will be sufficient.

We now analyze the running time of the algorithm. As argued above we only expect to test $O(1)$ values for e. The main complexity parts come from the lattice reduction algorithm, $O(\log p)$ pseudoprimality tests (on numbers that we heuristically assume to behave like random numbers of the same size), one execution of Cornacchia's algorithm and one execution of Algorithm 1. The lattice reduction in dimension 2 and Cornacchia's agorithm are both variants of the Euclidean algorithm and require $\tilde{O}(\log^2 p)$ bit operations. The primality tests can be done using Miller-Rabin; this requires $O(1)$ modular exponentiations for most random numbers and $O(\log p)$ modular exponentiations for prime numbers, in other words $\tilde{O}(\log^3 p)$ operations in total. This dominates the cost of Algorithm 3, as the cost of Algorithm 1 is less than that.

Note that paths of length about $3 \log_\ell p$ are expected to be necessary and sufficient for random matrices in G (since G is of size roughly $p^3/2$). We therefore expect that the factorizations we compute are close to optimal ones unless diagonal matrices are particularly close to the identity in LPS graphs, which there is no reason to believe a priori.

5 Improved algorithm in the general case

We now consider the case of arbitrary matrices in G. In [10] Petit et al. showed how to factor general matrices with four diagonal matrices, for a total factorization length of $16 \log_\ell p$. Here we factor general matrices with just two diagonal matrices and one matrix in Ω, for a total expected length of $(7+o(1)) \log p$.

Let $J := \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$. We will use the following lemma.

Lemma 7 *Let $a, b, c, d \in \mathbb{Z}$. We have*

1. $\left(\begin{smallmatrix} a+bi & c+di \\ -c+di & a-bi \end{smallmatrix}\right) = \left(\begin{smallmatrix} a+bi & 0 \\ 0 & a-bi \end{smallmatrix}\right) + \left(\begin{smallmatrix} c+di & 0 \\ 0 & c-di \end{smallmatrix}\right) J,$
2. $J \left(\begin{smallmatrix} a+bi & 0 \\ 0 & a-bi \end{smallmatrix}\right) = \left(\begin{smallmatrix} a-bi & 0 \\ 0 & a+bi \end{smallmatrix}\right) J.$

These properties are of course preserved by reduction modulo p through the homomorphism ϕ. For any diagonal matrix $D = \left(\begin{smallmatrix} a+bi & 0 \\ 0 & a-bi \end{smallmatrix}\right)$ we define its conjugate $\bar{D} = \left(\begin{smallmatrix} a-bi & 0 \\ 0 & a+bi \end{smallmatrix}\right)$, which is equal to D^{-1} up to a scalar factor.

Let $m \in G$. Our strategy is to find a matrix $m_2 \in \Omega$ and two diagonal matrices $m_1, m_3 \in G$ such that

$$m_1 m = \phi(m_2) m_3, \tag{5}$$

then to use the factorization algorithms for Ω elements and diagonal matrices to compute a factorization of m. (Note that since the inverse of any element in Σ is also in Σ, a factorization of m_1 will also give a factorization of m_1^{-1}.)

Following Lemma 7 we write $m = D + EJ$ with D, E diagonal matrices in G, and similarly we write $m_2 = X_2 + Y_2 J$, with X_2, Y_2 matrices of the form $\left(\begin{smallmatrix} w+xi & 0 \\ 0 & w-xi \end{smallmatrix}\right)$. We also define $X_1 := m_1$ and $X_3 := m_3$ to highlight that they are diagonal matrices. Equation 5 and the requirement that m_2 is in Ω then lead to the following system:

$$\begin{cases} X_1 D = \phi(X_2) X_3, \\ X_1 E = \phi(Y_2) \overline{X_3}. \end{cases} \tag{6}$$

Taking determinants of both equations and dividing one by another, we deduce

$$\frac{\det D}{\det E} = \frac{\det X_2}{\det Y_2} \bmod p.$$

Since we require $m_2 \in \Omega$ we also have

$$\det X_2 + \det Y_2 = \ell^e$$

for some positive integer e. These last two equations together fix the determinants of X_2 and Y_2 up to a multiple of p, for each value of e:

$$\det X_2 = \frac{\det D}{\det m} \cdot \ell^e \bmod p, \qquad \det Y_2 = \frac{\det E}{\det m} \cdot \ell^e \bmod p. \tag{7}$$

Given the determinant d of a matrix of the form $\left(\begin{smallmatrix} w+xi & 0 \\ 0 & w-xi \end{smallmatrix}\right)$, we can recover suitable values for w and x by solving a Diophantine equation $w^2 + x^2 = d$; this can be done efficiently as long as a factorization of d is known and all prime factors congruent to 3 modulo 4 appear an even number of times.

In our algorithm (Algorithm 4 below), we will fix e a priori to $\lceil \log_\ell p \rceil$, then try every possible value of $\det X_2 < \ell^e$ consistent with its value modulo p, and deduce $\det Y_2 = \ell^e - \det X_2$, until both $\det X_2$ and $\det Y_2$ are a power of 2 times a prime congruent to 1 modulo 4. As in previous algorithms, this restriction may lead to suboptimal factorization lengths but it allows to avoid a costly factorization step. Once the condition is satisfied we apply Cornacchia's algorithm on both $\det X_2$ and $\det Y_2$ to obtain X_2 and Y_2, and we deduce the matrix m_2. If no suitable values for $\det X_2$ and $\det Y_2$ are found we simply increase e and try again.

Let us now assume that X_2 and Y_2 are fixed as above. From System 6 we deduce

$$\begin{cases} X_1 = \phi(X_2) X_3 D^{-1} = \phi(Y_2) \overline{X_3} E^{-1} \\ X_3 = \phi(X_2)^{-1} X_1 D = \overline{E}\ \overline{X_1} \phi(\overline{Y_2})^{-1} \end{cases} \tag{8}$$

As diagonal matrices commute, each of these two equations can be rewritten in the form

$$X_i K_i = \overline{X_i}$$

for $i = 1, 3$ and some known diagonal matrices K_i. Moreover by the way we constructed X_2 and Y_2 we have $\det K_i = 1$.

If K_i is the identity then we can take the identity for X_i. Otherwise we write $X_i = \left(\begin{smallmatrix} w+x\iota & 0 \\ 0 & w-x\iota \end{smallmatrix}\right)$ and $K_i = \left(\begin{smallmatrix} w_0+x_0\iota & 0 \\ 0 & w_0-x_0\iota \end{smallmatrix}\right)$ for unknown w, x and known w_0, x_0, and we deduce the following system

$$\begin{pmatrix} w_0 - 1 & -x_0 \\ x_0 & w_0 + 1 \end{pmatrix} \begin{pmatrix} w \\ x \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Moreover since $\det \left(\begin{smallmatrix} w_0-1 & -x_0 \\ x_0 & w_0+1 \end{smallmatrix}\right) = w_0^2 + x_0^2 - 1 = \det K_i - 1 = 0$, this system is equivalent to the single equation

$$(w_0 - 1)w - x_0 x = 0$$

where $w_0 - 1$ and x_0 are both non zero, as otherwise K_i would be the identity. We can therefore choose $w = x_0$ and $x = w_0 - 1$, in other words

$$K_i = \left(\begin{smallmatrix} x_0+(w_0-1)\iota & 0 \\ 0 & x_0-(w_0-1))\iota \end{smallmatrix}\right).$$

In order to call our diagonal matrix factorization algorithm on X_1 and X_3 we also need their determinants to be quadratic residues modulo p. By assumption $\det m$ and $\det m_2 = \ell^e$ are quadratic residues, so either both $\det X_1$ and $\det X_3$ are quadratic residues or none of them is. In our algorithm when they are non quadratic residues we just search for new values of $\det X_2$ and $\det Y_2$.

In addition to the heuristic involved in Lemma 6, Algorithm 4 requires other heuristics related to the probability to find suitable determinants for X_2 and Y_2 that are not too large, and the probability that both diagonal matrices have square determinant. Under these heuristics the runtime complexity and the output of this algorithm are given by the following lemma:

Lemma 8 *Let $m \in G$. Then under plausible heuristic assumptions Algorithm 4 requires $\tilde{O}(\log^4 p)$ bit operations and it returns factorizations of lengths $(7 + o(1)) \log_\ell p$.*

Proof Taking as definition of "Cornaccia-nice" that the integers are pseudo-primes congruent to 1 modulo 4, the condition is satisfied for both n_X and n_Y with a probability $1/16 \log^2 p$ (here we heuristically assume that these numbers behave like random numbers of the same size). Using Miller-Rabin pseudo-primality test, we will need one modular exponentiation on most couples, and $O(\log p)$ modular exponentiations when at least one of the two numbers is prime. In total we will need $O(\log^2 p)$ modular exponentiation until we have a couple (n_X, n_Y) of the correct form, which amount to $\tilde{O}(\log^4 p)$ bit operations. Cornacchia's algorithm itself is more efficient than that, as are all the other parts of the algorithm. Note that to allow enough randomness in the choice of n_X and n_Y we expect e to grow up to roughly $\log_\ell p + \log\log_\ell p$, so we expect the while loop to be executed only $O(\log\log p)$ times. The expected length is e plus twice the expected length for Algorithm 3.

Algorithm 4 General factorization algorithm

Require: matrix $m = \left(\begin{smallmatrix} A+Bi & C+Di \\ -C+Di & A-Bi \end{smallmatrix}\right) \in G$

Ensure: list of factors $L = (s_1, \ldots, s_N)$ such that $s_i \in S$ and $m = \prod_{i=1}^{N} s_i$

$e \leftarrow \lceil \log_\ell p \rceil$
$n_x \leftarrow \ell^e(a^2+b^2)/(a^2+b^2+c^2+d^2)$
$n_X \leftarrow$ smallest positive integer congruent to n_x modulo p and to 1 modulo 4
while $n_X \leq \ell^e$ **do**
  $n_Y \leftarrow \ell^e - n_X$
  if both n_X and n_Y are "nice for Cornacchia's algorithm" (for example they
    are power of four times a (pseudo)prime congruent to 1 mod 4) **then**
    $X_2 \leftarrow \left(\begin{smallmatrix} w+xi & 0 \\ 0 & w-xi \end{smallmatrix}\right)$ where $w^2+x^2 = n_X$
    $Y_2 \leftarrow \left(\begin{smallmatrix} y+zi & 0 \\ 0 & y-zi \end{smallmatrix}\right)$ where $y^2+z^2 = n_Y$
    $m_2 \leftarrow \left(\begin{smallmatrix} w+xi & y+zi \\ -y+zi & w-xi \end{smallmatrix}\right)$
    $K_1 \leftarrow \phi(X_2)^{-1}\phi(\overline{Y_2})D\overline{E^{-1}}$
    $K_3 \leftarrow \phi(X_2)\phi(Y_2)^{-1}D^{-1}E$
    for $i = 1, 3$ **do**
      if $K_i = I$ **then**
        $X_i \leftarrow I$
      else
        $X_i \leftarrow \left(\begin{smallmatrix} x_0+(w_0-1)\iota & 0 \\ 0 & x_0-(w_0-1))\iota \end{smallmatrix}\right)$ where $\left(\begin{smallmatrix} w_0+x_0\iota & 0 \\ 0 & w_0-x_0\iota \end{smallmatrix}\right) = K_i$
      end if
    end for
    if $\det X_1$ is a square **then**
      Run Algorithm 3 on X_1^{-1} to obtain a list L_1
      Run Algorithm 1 on $X_2 + Y_2 J$ to obtain a list L_2
      Run Algorithm 3 on X_3 to obtain a list L_3
      Replace every element s in L_2 by its projection $\phi(s)$
      Concatenate the three lists in a single list L
      return L
    end if
  else
    $n_X \leftarrow n_X + 4p$
  end if
end while
$e \leftarrow e + 1$
go to Step 2

Note that paths of length about $3\log_\ell p$ are expected to be necessary and sufficient for random matrices in G. The factorizations produced by our algorithm are therefore expected to be a factor 7/3 larger than optimal ones, instead of 16/3 larger than optimal ones for the algorithm of Petit et al. [10]. Moreover, the best diameter bound on LPS graphs is $6\log p$ so in that sense we can expect that our factorizations are only 16% larger than optimal ones in the worst case.

6 Experimental Validation

Because of the heuristic nature of our analysis we have experimentally checked our results using the computer algebra package Magma [16]. In our experiments we fixed $\ell = 5$ and we generated various p such that $p = 1 \bmod 4$ and $\left(\frac{\ell}{p}\right) = 1$. For each such set of parameters we generated various matrices of square determinant, and we ran Algorithm 4 on them. The resulting factorization lengths are displayed in Figure 6 as functions of the parameter $\log_\ell p$. The best near square approximation line for the data points has a slope equal to 7.03, confirming our analysis that the lengths are $(7 + o(1))\log_\ell p$.

7 Conclusion

We presented a new algorithm that computes short paths between arbitrary pair of vertices in LPS Ramanujan graphs. Subject to several plausible heuristics the algorithm is polynomial time and for LPS graphs with parameters p and ℓ it returns paths of length $(7 + o(1))\log_\ell p$.

Acknowledgements This work was done while the second author was working at the University of Oxford under a research grant from the UK government.

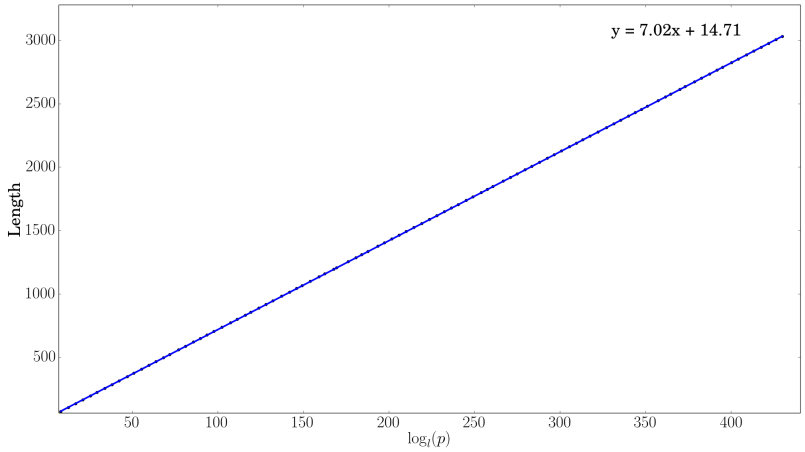

Fig. 1 Factorization lengths of our algorithm for general matrices.

References

1. Alex Bocharov Andreas Blass and Yuri Gurevich. Optimal ancilla-free pauli+v circuits for axial rotations, 2014.
2. D. Charles and K. Lauter. Computing modular polynomials, 2005.
3. Denis Xavier Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009.
4. G. Cornacchia. Su di un metodo per la risoluzione in numeri interi dell' equazione $\sum_{h=0}^{n} c_h x^{n-h} y^h = p$. *Giornale di Matematiche di Battaglini*, 46:33–90, 1903.
5. Giuliana Davidoff, Peter Sarnak, and Alain Valette. *Elementary Number Theory, Group Theory, and Ramanujan Graphs*. Cambridge University Press, 2003.
6. Harald Andrés Helfgott. Growth and generation in $SL_2(Z/pZ)$. *Ann. of Math. (2)*, 167 (2):601–623, 2008.
7. Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43:439–561, 2006.
8. A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.
9. Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*. Springer, 2002.
10. Christophe Petit, Kristin Lauter, and Jean-Jacques Quisquater. Full cryptanalysis of LPS and Morgenstern hash functions. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2008.
11. Neil J. Ross. Optimal ancilla-free clifford+v approximation of z-rotations, 2015.
12. Naser T. Sardari. Complexity of strong approximation on the sphere. `arxiv.org/abs/1703.02709`, 2017.
13. Naser T. Sardari. The least prime number represented by a binary quadratic form. arxiv.org/abs/1803.03218, 2018.
14. Peter Sarnak. *Some Applications of Modular Forms*. Cambridge University Press, 1990.
15. Jean-Pierre Tillich and Gilles Zémor. Collisions for the LPS expander graph hash function. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 254–269. Springer, 2008.
16. C. Fieker A. Steel (eds.) W. Bosma, J. J. Cannon. Handbook of Magma functions, edition 2.20. `http://http://magma.maths.usyd.edu.au/magma/`, 2013.